ElGamal homomorphic encryption: $PP = (p, g)$

Multiplicatively homomorphic encryption.
We need an additively-multiplicative encryption.

$n_1 = g^{m_1} \bmod p \Rightarrow Enc(PuK_A = a, i_1, n_1) = (E_1, D_1) =$
$= (E_1 = n_1 \cdot a^{i_1} \bmod p,\ D_1 = g^{i_1} \bmod p).$

$n_2 = g^{m_2} \bmod p \Rightarrow Enc(PuK_A = a, i_2, n_2) = (E_2, D_2) =$
$= (E_2 = n_2 \cdot a^{i_2} \bmod p,\ D_2 = g^{i_2} \bmod p.$

B: $PrK_B = y$; $PuK_B = b$.
$b = g^y \bmod p$

A: $PrK_A = x$; $PuK = a$.
$a = g^x \bmod p$

$PuK_A = a$. n - message $r \leftarrow$ randi
$Enc(a, r, n) = c = (E, D) \longrightarrow$ A: $Dec(x, c) = n$
$E = n \cdot a^r \bmod p$; $D = g^r \bmod p$
1) $D^{-x} \bmod p$
2) $E \cdot D^{-x} \bmod p = n$

$$Enc: \mathcal{I}_{p-1}^{r} \times \mathcal{I}_p^{n} \longleftrightarrow \mathcal{I}_p^{E} \times \mathcal{I}_p^{D}$$

Let n_1, n_2 - are messages to be encrypted: $n_1 \cdot n_2 \bmod p = n_{12}$
r_1, r_2 - are random numbers for encryption

$$\left.\begin{array}{l} Enc(a, r_1, n_1) = c_1 = (E_1, D_1) \\ Enc(a, r_2, n_2) = c_2 = (E_2, D_2) \end{array}\right\}$$

$Enc(a, (r_1 + r_2) \bmod (p-1), n_1 \cdot n_2 \bmod p) =$

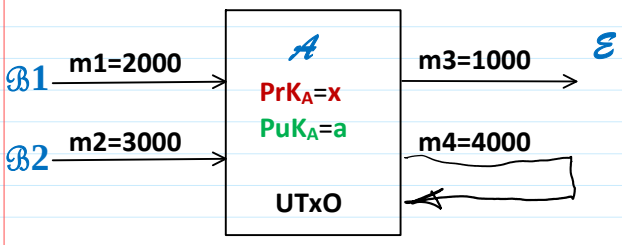
$= c_1 * c_2 = c_{12} = (E_1 * E_2 \bmod p,\ D_1 * D_2 \bmod p) = (E_{12}, D_{12})$

$Dec(x, c_{12}) = n_1 \cdot n_2 \bmod p = n_{12}$.

Transaction - **Tx** in blockchain

Balance: $m_1 + m_2 = m_3 + m_4$

Transaction - **Tx** in blockchain



Balance: $m_1 + m_2 = m_3 + m_4$

How to verify balance when inputs and expenses are encrypted?

Assume that $n_1 = g^{m_1} \bmod p$; $n_2 = g^{m_2} \bmod p$ ➡ $n_1 * n_2 = g^{m_1+m_2} \bmod p$

$n_3 = g^{m_3} \bmod p$; $n_4 = g^{m_4} \bmod p$ ➡ $n_3 * n_4 = g^{m_3+m_4} \bmod p$

Since DEF: $\mathcal{Z}_{p-1} \leftrightarrow \mathcal{Z}_p^*$ is 1-to-1 (bijective) then

if $(m_1 + m_2) \bmod (p-1) = (m_3 + m_4) \bmod (p-1)$

then $n_1 \cdot n_2 \bmod p = n_3 \cdot n_4 \bmod p$

If $m_1 + m_2 < p-1$ & $m_3 + m_4 < p-1$

And $m_1 + m_2 = m_3 + m_4$ ⬌ $n_1 \cdot n_2 \bmod p = n_3 \cdot n_4 \bmod p$

A: $B_1 \xrightarrow{2000} A \xrightarrow{4000} E$

$B_2 \xrightarrow{3000} A \xrightarrow{4000}$

$Enc(a, r_1, n_1) = c_1$ } $Enc(a, r_2, n_2) = c_2$ }

$Enc(a, r_3, n_3) = c_3$ } $Enc(a, r_4, n_4) = c_4$ }

Net

$c_{12} = c_1 \cdot c_2 \bmod p$ & $c_{34} = c_3 \cdot c_4 \bmod p$

Since DEF: $\mathcal{Z}_{p-1} \leftrightarrow \mathcal{Z}_p^*$ is 1-to-1 (bijective) then

If $(m_1 + m_2) \bmod (p-1) = (m_3 + m_4) \bmod (p-1)$

$c_{12} = c_{34}$

AA - Audit Authority $PrK_{AA} = z$ $PuK_{AA} = d = g^z \bmod p$

A instead of encryption with her $PuK = a$

she encrypts n_1, n_2 and n_3, n_4 with $PuK_{AA} = d$.

Then the same verification property is valid to verify transaction

she encrypts n_1, n_2 and n_3, n_4 with $PuK_{AA} = d$.
Then the same verification property is valid to verify transaction balance

AA can decrypt c_1, c_2 and c_3, c_4 and to verify ballance

n_1, n_2 $\quad$ n_3, n_4

How to compute m_1, m_2 $\quad$ m_3, m_4 if

$n_1 = g^{m_1} \bmod p$; $n_2 = g^{m_2} \bmod p$; $n_3 = g^{m_3} \bmod p$; $n_4 = g^{m_4} \bmod p$

If, e.g. $a = g^x \bmod p$ it is ifeasible to find exponent x, when g, p, a are given.

On this is relied the security of El-Gamal, Schnorr and etc. cryptographic systems.

The problem to find x is named as Discrete Logarithm Problem (DLP).

To solve DLP is infeasible if p, x are sufficiently large:

$p \sim 2^{2048} \approx 10^{700}$ & $x \sim 2^{2048} \approx 10^{700}$

$|p| \leq 2048$ bits $\quad$ $|x| \leq 2048$ bits

E.g. by having $n_1 = g^{m_1} \bmod p$ when $m_1 \leq 1\,000\,000 = 10^6$

Sum m_1 can be found by total scan from relation $n_1 = g^{m_1} \bmod p$ from $m_1 = 1$ up to $m_1 = 10^6$ and find m_1 satisfying

Homomorphic CryptoSystems: Computation with encrypted data in Data Center.

Declare **Public Parameters** to the network **PP** = (***p***, ***g***); $\quad$ ***p*= 268435019**; ***g*=2**;
In real cryptosystem is chosen having 2048 bits and is of order ***p***=$2^{2048} \sim 10^{700}$, i.e. |***p***|=2048 bits.
In our simulation we use |***p***|=2048 bits, i.e. ***p*** < 2^{28} = 268 435 456.

$PrK_A = x$; $PuK_A = a$: $a = g^x \bmod p$.

Received encrypted incomes I_1, I_2 from B_1, B_2

B_1 : $I_1 = 2000 \rightarrow n_1 = g^{I_1} \bmod p \rightarrow Enc(a, i_1, n_1) = c_{1a} = (E_{1a}, D_{1a})$

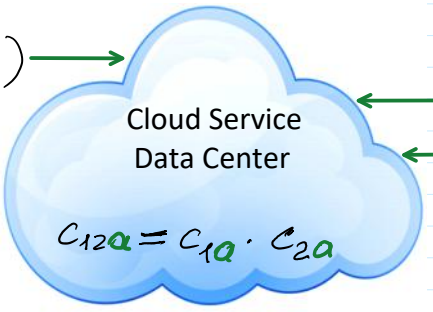
B_2 : $I_2 = 3000 \rightarrow n_2 = g^{I_2} \bmod p \rightarrow Enc(a, i_2, n_2) = c_{2a} = (E_{2a}, D_{2a})$

Total Income of Alice is: $I_{12} = I_1 + I_2 \bmod (p-1)$

Since $I_1 + I_2 < p-1$, then $I_1 + I_2 \bmod (p-1) = I_1 + I_2$

Query (Total Incomes) →

$c_{12}a = (E_{12}a, D_{12}a)$ ←



c_1a ←

c_2a ←

$Dec(x, c_{12}a) = n_{12}$

$n_{12} = n_1 \cdot n_2 \bmod p = g^{I_1} g^{I_2} \bmod p = g^{I_1 + I_2} \bmod p = g^{I_{12}} \bmod p$

Having n_{12} Alice finds I_{12} by the search procedure from the equation: $n_{12} = g^{I_{12}} \bmod p$: $I_{12} = 100, 200, 300, \ldots, 5000$.

Till this place

$p = 11$; $g = 2$:

$m_1 = 1$; $g^{m_1} \bmod p$ → c_1 → $Dec(x, g^{m_1}) = g^{m_1} = n_1$

$m_2 = 2$; $g^{m_2} \bmod p$ → c_2 → $Dec(x, g^{m_2}) = g^{m_2} = n_2$

$(m_1 + m_2) \bmod (p-1) = (1+2) \bmod (p-1) = 3$

mod 10 mod 10

$m_1 = 4$, $m_2 = 9$ } $(m_1 + m_2) \bmod 10 = (4+9) \bmod 10 = 13 \bmod 10 = 3$

Prevention: $m_1 + m_2 < (p-1)/2$

I_{p-1}	0	1	2	3	4	5	6	7	8	9
						±5	-4	-3	-2	-1

Let transaction data must be declared to Audit Organization (AA) having key pait=r

$PrK_{AO}=z$; $PuK_{AO}=e$.

```
>> p=int64(268435019)          >> m1=2000;
p = 268435019                  >> m2=3000;
>> g=2;                        >> m12=mod(m1+m2,p-1)          % m1+m2=m12=5000 << (p-1)/2
```

```
>> p=int64(268435019)
p = 268435019
>> g=2;
>>
>> z=int64(randi(p-1))
x = 141076898
>>  a=mod_exp(g,x,p)
e = 116336486
```

```
>> m1=2000;
>> m2=3000;
>> m12=mod(m1+m2,p-1)          % m1+m2=m12=5000 << (p-1)/2
m12 = 5000
>> n1=mod_exp(g,m1,p)
n1 = 28125784
>> n2=mod_exp(g,m2,p)
n2 = 222979214
```

m_1 : $i_1 \leftarrow randi(\mathbb{Z}_{p-1})$

E1 = $\mathcal{E}_1 = n_1 * e^{i_1} \bmod p$

D1 = $\delta_1 = g^{i_1} \bmod p$

```
>> i1=int64(randi(p-1))
i1 = 256575903
>> e_i1=mod_exp(e,i1,p)
a_i1 = 177744290
>> E1=mod(n1*e_i1,p)
E1 = 78907012
>> D1=mod_exp(g,i1,p)
D1 = 71219017
```

Computations in Cloud data base

```
>> E12=mod(E1*E2,p)
E12 = 248852506
>> D12=mod(D1*D2,p)
D12 = 220753507
```

c12=(E12, D12)

m_2 : $i_2 \leftarrow randi(\mathbb{Z}_{p-1})$

E2 = $\mathcal{E}_2 = n_2 * e^{i_2} \bmod p$

D2 = $\delta_2 = g^{i_2} \bmod p$

```
>> i2=int64(randi(p-1))
i2 = 148753825
>> a_i2=mod_exp(e,i2,p)
e_i2 = 206019372
>> E2=mod(n2*e_i2,p)
E2 = 144070332
>> D2=mod_exp(g,i2,p)
D2 = 20873398
```

Audit Organization decrypts c12 =(E12, D12)

```
>> mz=mod(-z,p-1)        % mines z mod p-1 computation
mz = 127358120
>> mod(mz+z,p-1)         % verification that mz+z mod(p-1)=0
ans = 0
>> D12_mz=mod_exp(D12,mz,p)   % D12^-z computation for decryption
D12_mz = 21824811
>> nnn12=mod(E12*D12_mz,p)    % decryption formula E12* D12^-z mod p = n12 nnn12
nnn12 = 143845522
```

Verification

```
>> nn12=mod(n1*n2,p)
nn12 = 143845522
>> n12=mod_exp(g,m12,p)
n12 = 143845522
```

$n_1 * n_2 = g^{m_1+m_2} \bmod p$

dlog.m

```
% Finds discrete logarithm value corresponding to exponent value i
% by total scan of i from start by step until fin
% p - is a strong prime (Public Parameter)
% g - is a generator (Public Parameter)
% def - is a discrete exponent function value computed by mod_exp(g,i,p)
%       where dl=i is a searchable value of exponent
%
function dl = dlog(p, g, def, start, step, fin)
 dl=0;
 i=start;
 while i<fin
    ee=mod_exp(g,i,p);
```

```
>> def=nnn12
def = 143845522
>> start=0
start = 0
>> step=100
step = 100
```

```
 while i<fin
    ee=mod_exp(g,i,p);
    if ee==def
      dl=i;
      return;
    endif
    i+=step;
 endwhile
 disp('Exponent is not found!');
end
```

```
start = 0
>> step=100
step = 100
>> fin=9900
fin = 9900
>>
>> dl = dlog(p, g, def, start, step, fin)
dl = 5000
```